НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ім. ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Литвиненко К. А.
Завідувач кафедри
_____ Олександр Коваль
« __»__ _____2020 р.

Дипломна робота

на здобуття ступеня бакалавра
з напряму підготовки 122 «Комп'ютерні науки та інформаційні технології»
на тему: «Проектування та розроблення WEB – додатків на платформі систем контролювання доступу автомобільного транспорту "CarGo Enterprise"»

Виконав (-ла):
студент (-ка) IV курсу, групи ТМ-61
Литвиненко Костянтин Андрійович _____

Керівник:
Професор кафедри АПЕПС, к.е.н.
Сігайов Андрій Олександрович _____

Рецензент:
Ст. викладач кафедри АПЕПС, к.т.н.
Воробйов Микита Валерійович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Інформаційні технології моніторингу довкілля

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ

на дипломну роботу студенту

Литвиненку Костянтину Андрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Проектування та розроблення WEB – додатків на платформі систем контролювання доступу автомобільного транспорту “CarGo Enterprise”

керівник роботи Сігайов А. О. професор кафедри АПЕПС, кандидат економічних наук
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. №

2. Строк подання студентом роботи 30 Вересня 2019

3. Вихідні дані до роботи _____

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

проаналізувати існуючі програмні рішення та можливі засоби реалізації задачі, обґрунтувати обрані програмні застосунки та шляхи розробки програмних додатків, створити програмне забезпечення, розробити користувацький інтерфейс, зробити висновки за результатами роботи

_____

5. Перелік ілюстративного матеріалу

_____

_____

_____

_____

_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «30 Вересня 2019р».

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	30.09.2019	
2.	Вивчення та аналіз задачі	25.11.2019	
3.	Розробка архітектури та загальної структури системи	6.01.2020	
4.	Розробка структур окремих підсистем	1.02.2020	
5.	Програмна реалізація системи	1.03.2020	
6.	Оформлення пояснювальної записки	1.05.2020	
7.	Захист програмного продукту		
8.	Передзахист	08.06.2020	
9.	Захист	17.06.2020	

Студент _____ Литвиненко К. А.____
(підпис) (прізвище та ініціали,)

Керівник роботи ____ _____ ____Сігайов А. О._____
(підпис) (прізвище та ініціали,)

ВІДГУК

керівника дипломної роботи

освітньо-кваліфікаційного рівня „бакалавр”

виконаної на тему : Проектування та розроблення WEB додатків на платформі систем контролювання автомобільного транспорту «CarGo Enterprise»

студентом групи ТМ-61, Литвиненко Костянтином Андрійовичем

(прізвище, ім’я, по батькові)

Дипломна робота спорямована на створення WEB-додатку для автоматизованої системи контролю пропускної здатності автостоянки.

Щодо основних етапів реалізації роботи необхідно підкреслити, що в роботі детально проаналізовано теоретичний матеріал, який дозволив висвітлити шляхи вирішення поставленої задачі. Під час виконання роботи студент проявив використання творчої складової в роботі та продемонстрував гарну роботу в команді. Реалізація програмного продукту була виконана на сучасних технологіях проектування було застосовано технології. Базовою концепцією при проектуванні інформаційної системи було забезпечення масштабованості, гнучкості та надійності програмного продукту з можливістю подальшого удосконалення або додавання функціоналу.

Робота виконана у повному обсязі та відповідає завданню і вимогам, що Литвиненко К.А. заслуговує на присвоєння кваліфікації бакалавра комп’ютерних наук та інформаційних технологій з напряму підготовки 122 Комп’ютерні науки та інформаційні технології за спеціалізацією Інформаційні технології моніторингу довкілля.

Керівник дипломної роботи

професор, к.е.н.		Сігайов А.О.
(посада, вчені звання, ступінь)	(підпис)	(ініціали, прізвище)

РЕЦЕНЗІЯ

на дипломну роботу
на здобуття ступеня бакалавра

виконану на тему: *Проектування та розроблення WEB – додатків на платформі систем контролювання доступу автомобільного транспорту "CarGo Enterprise"*

студентом (-кою) *Литвиненко Костянтин Андрійович*
(прізвище, ім'я, по батькові)

Студент групи ТМ-61 Литвиненко Костянтин Андрійович виконав дипломну роботу бакалавра на тему «Проектування та створення WEB додатків на платформі систем контролювання автомобільного транспорту «CarGo Enterprise». В ході виконання дипломної роботи було розроблено автоматизовану систему контролю доступу автомобільного транспорту на територію автостоянки.

Під час виконання дипломної роботи студент проявив уміння аналізувати літературні джерела, приймати правильні технічні рішення, застосовувати сучасні системні та інформаційні технології, обробляти та аналізувати результати розрахунків.

Атестаційна робота виконана студентом самостійно відповідно до завдання.

В цілому, робота бакалавра виконана на достатньому рівні, не має елементів плагіату та заслуговує оцінки «задовільно» а автор бакалаврської роботи Литвиненко К.А. заслуговує на присвоєння кваліфікації бакалавра з комп'ютерних наук за спеціальністю 122 "Комп'ютерні науки".

Рецензент

старший викладач к.т.н		Воробйов М.В.
(посада, вчені звання, ступінь)	(підпис)	(ініціали, прізвище)

Печатка установи, організації рецензента *(тільки для зовнішнього рецензента)*

АНОТАЦІЯ

Метою даної роботи є створення простого та зручного WEB-додатку для автоматизованого контролю доступом на територію автостоянки.

Прототипом для додатку слугує програма компанії Intteks «Cargo Parking Enterprise». База даних містить інформацію про всі автомобілі, які є на цій території на даний момент, та всі, які перебували.

Загальний обсяг роботи: 50 сторінок, 7 ілюстрацій та 13 посилань.

Ключові слова: автоматизована система, тарифікатор-парковка, база даних, WEB-додаток, автоматизоване робоче місце.

ABSTRACT

The purpose of this work is to create a simple and convenient WEB-application for automated access control to the parking lot.

The prototype for the application is the program of Intteks "Cargo Parking Enterprise". The database contains information about all the cars that are in the area at the moment, and all that were.

Total volume of work: 62 pages, 7 illustrations and 13 links.

Keywords: automated system, tariff-parking, database, WEB-application, automated workplace.

ЗМІСТ

Перелік умовних позначень, скорочень та термінів

1) АРМ – автоматизоване робоче місце
2) СКБД – система керування базами даних
3) БД – база даних
4) Frontend – фронтенд, або зовнішній вигляд програми
5) Backend – програмно-апаратна частина сервісу
6) API – опис способів, якими одна комп'ютерна програма може взаємодіяти з іншою програмою

ВСТУП

У сучасному світі дуже багато розробок спрямовано саме на автоматизацію роботи різноманітних систем, так як більшість видів роботи може бути виконана комп’ютерами. Подібні системи мають бути максимально простими у використанні та експлуатації.

Дана дипломна робота спрямована на розробку WEB додатку для автоматизованого контролю пропускної здатності транспорту на територію автостоянки. Додаток призначений для громадських автостоянок, власних та службових парковок, часних будинків тощо.

Основні цілі створення такого додатку – зменшення затрат на людську робочу силу та автоматизацію тарифікації вартості послуг. Не складний WEB додаток має переваги перед людською неточністю: автоматична ідентифікація машин, автоматичний підрахунок вартості. WEB додаток зчитує з камер номерні знаки машин та заносить до бази даних час прибуття, номера транспорту і тариф за використання.

WEB додаток має можливості: зчитування номерних знаків з камер відеоспостереження, занесення даних про водія та машину до бази даних. База даних містить повну історію користування парковкою, дані про клієнтів та панель редагування кожним клієнтом.

1. Дослідження та аналіз предметної област

За багато років розвитку люди зрозуміли, що людська робота та неточність можуть бути замінені роботою комп'ютера. Тому даний додаток представляє собою просте АРМ для контролю доступу і тарифікацію автостоянки. Завдяки максимальній простоті у використанні та експлуатації даний додаток може використовуватись будь ким та може бути легко та дешево введений на підприємство. У даному розділі розкрита сутність дипломного завдання та обумовлені задачі, що мали бути виконані у процесі виконання.

1.1. Мета створення додатку для контролю доступу до парковки

На даний момент існує чимала кількість парковок без автоматизованого місця оператора, де використовується застаріле обладнання для ведення історії обслуговування парковки, а підрахунки щодо вартості простою на території стоянки робляться власноруч оператором/охоронцем.

Метою цієї дипломної роботи є створення зручного автоматизованого робочого місця для тарифікації вартості за використання послуг автостоянки чи парковки та ведення історії надання послуг.

Дану задачу можна розбити на такі етапи роботи:

— створення додатку для перегляду;

— створення WEB-серверу, який буде обробляти дані;

— створення бази даних для подальшого заповлення при експлуатації системи;

— розробка програмного продукту та на писання коду.

Програмний продукт передбачає фунціонал для відображення інформації про машини та їх володарів. Завдяки йому користувач зможе побачити список усіх

клієнтів на даний момент на парковці та список усіх клієнтів за весь час роботи автостоянки. Програмний продукт повинен:

— ідентифікувати машину, що заїхала до парковки;

— Занести до бази даних інформацію;

— Порахувати тариф за використання парковки;

— Вивести тариф на екран на момент виїзду машини для оплати.

1.2. Вхідні дані

Розроблена система автоматизованого контролю спрямована на зчитування даних з камер відеоспостереження, зображення з яких показується на сторінці WEB-додатку. Після зчитування та розпізнавання номерних знаків система може приступати до обробки даних.

1.3. Компоненти системи

У даній розробці дипломної роботи можна виділити головні частини, що забезпечують працездатність системи:

— база даних;

— web-додаток, який показує в’їзд та відображає клієнтів.

— web-сервер який підраховує тариф

Розроблене АРМ надає користувачеві можливість легко та без зусиль контролювати пропускну здатність автостоянки.

1.4. Потенційні користувачі

Система, розроблена у ході виконання цієї бакалаврської дипломної роботи може використовуватись різними компаніями для власних парковок, міськими

автостоянками тощо. Система не вимагає у користувача спеціальних навичок володіння ПК, тому може бути використана людиною, що не користується комп’ютерами у повсякденному житті.

Програма може бути куплена приватним підприємством, державним урядуванням чи іншими юридичними особами, які потребують даний продукт.

ВИСНОВКИ ДО РОЗДІЛУ 1

Отже, в даному розділі ми розібрали доцільність створення системи контролю доступу до автостоянки. Оператор, який буде працювати з додатком, буде мати максимально зрозумілий та простий інструмент для роботи з пропускною здатністю до парковки. Таким чином, до роботи можна буде привернути людей з низькою кваліфікацією у роботі з ПК. База даних буде розміщего і зв’язана з сервером на поточному ПК задля зменшення затрат на необхідне обладнання та прискорення введення системи в експлуатацію. Дана система зменшить затрати на обслуговування парковки та підвищить надійність роботи всієї парковки вцілому.

2. ПОБУДОВА МОДЕЛІ ПРЕДМЕТНОЇ ОБЛАСТІ

В даній дипломній роботі буде використовуватись архітектура REST.

У загальному випадку REST - це дуже простий інтерфейс управління інформацією без використання каких-то додаткових внутрішніх прослоек. Інформація про однорідну інформацію однозначно визначає глобальний ідентифікатор, таким чином як URL.

За допомогою Representational State Transfer (* передача репрезентативного стану), або REST, описується архітектурний стиль для веб-сервісів. REST складається з ряду стандартів або обмежують вимог щодо обміну даними між різними системами, і для систем, які працюють згідно з принципами REST, використовується термін RESTful. REST - абстрактна концепція, це не мова, не фреймворк або тип програмного забезпечення.

Для кращого розуміння REST вам могла б допомогти аналогія, коли порівнюється колекція вінілових платівок з сервісом для потокової передачі аудіо-даних (* спосіб відтворення аудіо- та відеоматеріалів з інтернету без їх попереднього завантаження в комп'ютер). У випадку з віниловими платівками для їх поширення необхідно, щоб для кожного запису була створена повна копія. У випадку ж з сервісом для потокової передачі аудіо-даних та ж сама музика може поширюватися необмежений термін, якщо відомі будь-які дані, наприклад назву пісні. В цьому випадку сервіс для потокової передачі аудіо-даних є RESTful сервісом, а колекція вінілових платівок - немає.

API (Application Programming Interface) - інтерфейс, який дозволяє програмам спілкуватися один з одним. RESTful API - просто API, який працює згідно з принципами та поняттями REST. У випадку з API, які працюють в мережі, сервер отримує запит за допомогою кінцевої точки, де вказано URL-адресу, і відправляє назад відповідь, яким часто є дані в форматі на зразок JSON.

Система буде складатися с 3 основних частин:

- Сервер – модуль для зв'язку між базою даних та додатком, на якому будуть проходити всі обчислення.

- WEB-додаток – проста WEB-сторінка, за допомогою якою можна буде переглядати та редагувати дані про використання парковки, слідкувати за трансляцією з камер відеоспостереження, надавати різноманітні привілегії первим клієнтам та встановлювати тарифікацію за використання послугами.

- База даних – сховище інформації про всіх клієнтів, їх тарифи та історію відвідувань.

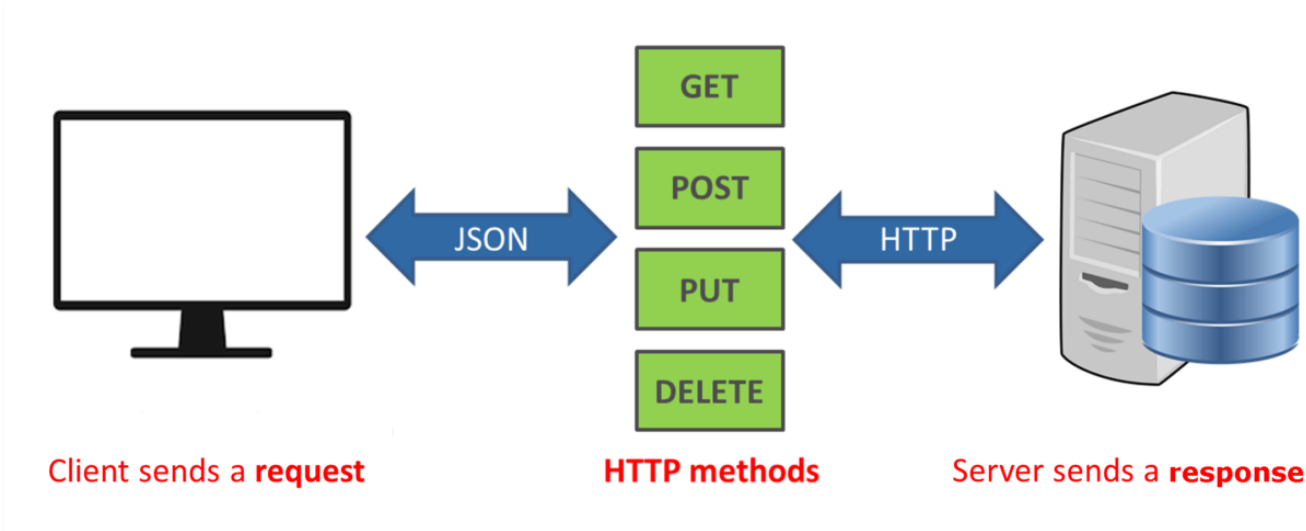

Рисунок 2.1.

Клієнт - це людина або програмне забезпечення, яке використовує API. Це може бути розробник, наприклад, ви, як розробник, можете використовувати API API для читання та запису даних із Twitter, створення нового твіту та виконання більше дій у програмі, яку ви пишете. Ваша програма дзвонить API Twitter. Клієнтом також може бути веб-браузер. Перейшовши на WEB-сайт Twitter, ваш браузер - це той клієнт, який дзвонить API Twitter і використовує повернені дані для відображення інформації на екрані.

Ресурс - ресурсом може бути будь-який об’єкт, про який API може надавати інформацію. Наприклад, в Instagram API, ресурсом може бути користувач, фотографія, хештег. Кожен ресурс має унікальний ідентифікатор. Ідентифікатором

може бути ім’я або число. В ситуації з даною роботою, ресурсом виступає база даних, з історією використання послуг парковки.

Структура системи складається с наступних модулів: WEB-додаток на персональному компьютері, база даних подій автостоянки та WEB-сервер (номерні знаки, час заїзду, вартість простою).

Таким чином ми отримуємо просту клієнт-серверну модель нашої системи.

2.1. Склад необхідного обладнання

В склад наобхідного обладнання входить:

- Виконавчі пристрої для обмеження вільного проїзду на територію автостоянки (шлагбауми/ворота), відеокамери для зчитування номерів машин

- ПК, який буде знаходитись в приміщенні охоронця, тобто на в’їзді/виїзді з території автостоянки, на якому буде розгорнуто серверну частину, WEB-додаток АРМ та адаптери зчитування номерів з машин

- ПК – сервер, на який встановлено СУБД

Виходячи з переліку компонентів, які входять до необхідного обладнання, ми можемо побудувати архітектуру нашого проекту. Розглядається два можливих варіанти. В одному з них все програмне забезпечення, база даних та додаток запущені та працюють на одному пк.

Приклад даної має недоліки у вигляді небезпеки одночасного виведення з строю усього програмного забезпечення в разі неполідки за ПК. Приклад даної архітектури можна спостерігати на рисунку 2.2.

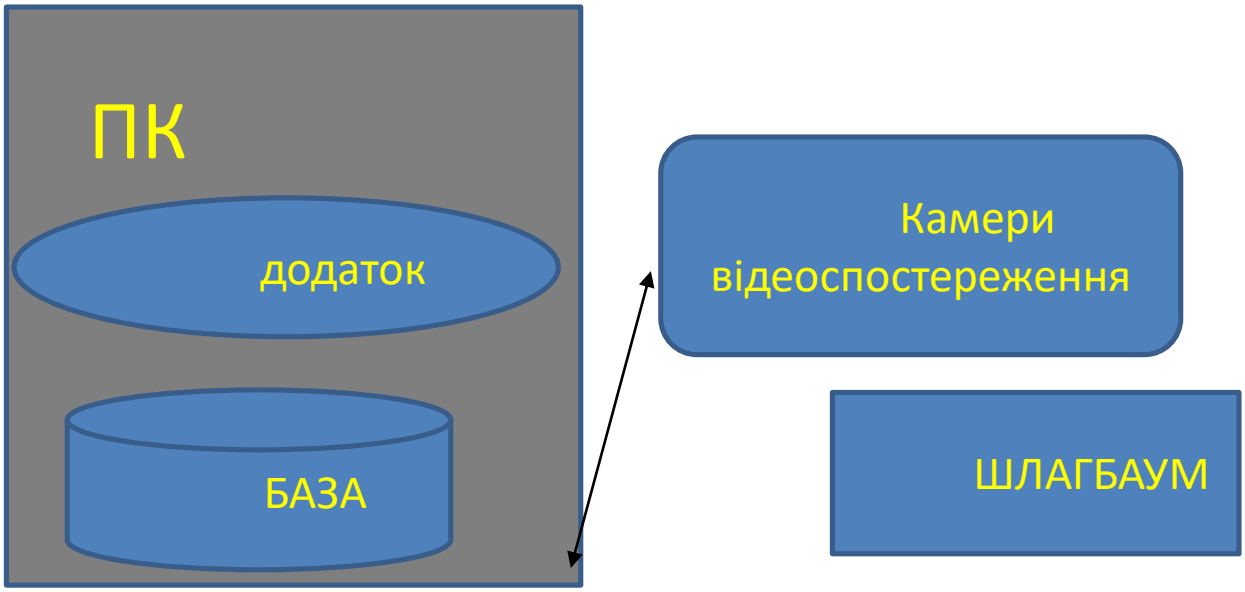

Рисунок 2.2.

Також є другий варіант, в якому оператор починає роботу з іншого пк або віддаленого робочого місця. У такому випадку база даних розгортається на іншому обладнанні і плюсом такої архітектури є зменшення ризиків втратити усі дані одночас з виведенням додатку з строю та зупитки роботи автоматизованого робочого місця.

Підключення відбувається за допомогою інтернет мережі між ПК з додатком та ПК з базою даних.

Приклад такої архітектури можна спостерігати на рисунку 2.3.

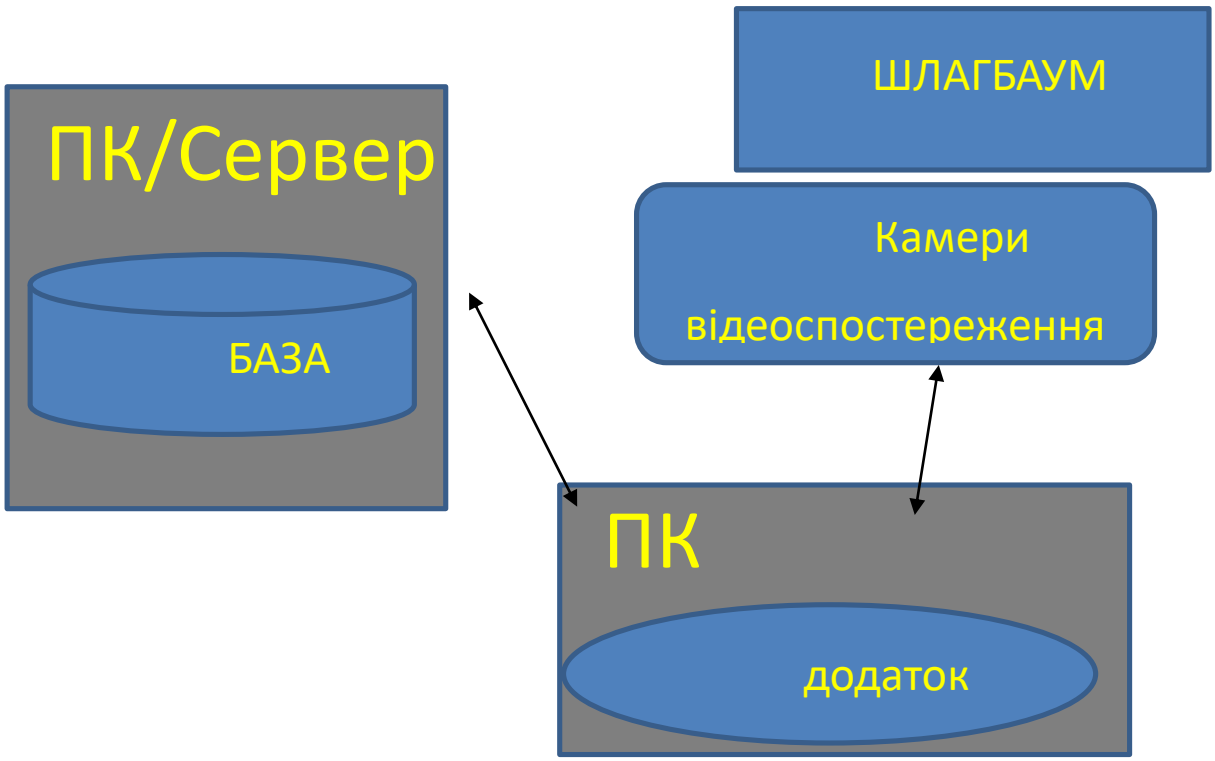

Рисунок 2.2.

2.2. Опис модуля «WEB-додаток»

Створення сайту з урахуванням ергономіки може бути визначено як здатність ефективно реагувати на потреби користувачів і забезпечувати їм комфорт при роботі з WEB-додатком.

Сайт має бути максимально простим та зрозумілим для використання. У загальному сенсі слова, термін ергономіка - це використання наукових знань про людину (психології, фізіології, медицини) з метою поліпшення умов роботи на робочому місці. Ергономіка в цілому характеризується двома принципами:

1) Комфорт під час використання, який складається з зменшення фізичної та психологічної втоми;
2) Безпека, яка передбачає вибір відповідних рішень для захисту користувача.

Додаток реалізує автоматизоване робоче місце (далі АРМ) оператора-завідуючого автостоянкою. Модуль є інтерактивним програмним засобом і може використовуватись для різних цілей:

-формувати політику тарифікації автомобілів

-формувати політику доступу до парковки

-відображати каталог автомобілів та клієнтів

-формувати звіти про функціонування автостоянки

2.3. Опис модуля «Сервер»

Серверна частина буде отримувати номерні знаки з відеокамер, та заносити відповідний час потрапляння машини на автостоянку до бази даних та встановлювати тариф для подальшого обчислення вартості за послуги.

Серверна частина може знаходитись на одному ПК з базою даних та WEB-додатком та по своїй суті являється його частиною. Також вона може знаходитись на іншому ПК задля більшої безпеки при аварійних ситуаціях.

2.4. Опис модуля «База даних»

База даних в моделі виступає у ролі сховища всієї інформації про перебування автомобілів на парковці.

База даних представляє собою структурований набір цифрової інформації, що зберігається у такому вигляді, що дає можливості для обробки автоматичними засобами. Головною характерною рисою баз даних є універсальність використання та збереження даних. Це означає, що зміна даних у БД не змушує модифікувати прикладне програмне забезпечення і навпаки.

Бази даних поєднують у собі такі функції:

— цілісність даних;

— повний доступ до інформації;

— створення залежності даних;

— швидкість доступу до інформації.

Для коректної та швидкої взаємодії з базами даних визначено дві концепції:

— система керування базами даних;

— адміністратор бази даних.

СКБД представляє собою програмний комплекс, що надає доступ до бази даних. Будь-яка СКБД може виконувати такі функції:

— створення таблиць і інших елементів БД;

— додавання записів до таблиці;

— редагування записів у таблиці;

— видалення записів, таблиць, інших елементів БД;

— пошук записів.

Система керування базами даних надає повний функціонал доступу до баз даних і спрощує взаємодію з ними. Реляційною базою даних являється БД, що

складається з взаємопов’язаних таблиць, кожна з яких містить дані про об’єкти певного типу [4].

Адміністратором бази даних виступає людина або група людей, в чиї обов’язки входить слідкувати за станом БД, функціями засобів керування БД та інтерфейсів.

Реляційні бази даних зберігають два типи даних при використанні в географічних інформаційних системах:

— атрибутивні (семантичні);

— графічні.

База даних буде містити в собі 2 основні таблиці:

- основна таблиця з історією використання послуг автостоянки за весь час, в якій буде відображатись ім’я, номерні знаки, дата та точний час приїзду транспорту, встановлений для неї тарифний план за використання послуг (таблиця 2.4.1.)
- таблиця, яка буде містити дані, про автомобільний транспорт, який знаходиться на території автостоянки в даний момент. В цій таблиці відсутній час виїзду транспорту з автостоянки.

Усі записи з другої таблиці потрапляють до першої в момент виїзду машини з території автостоянки. В залежності від тарифного плану та кількості проведеного часу на території автостоянки, в першу таблицю записується вартість за простій.

Журнал автостоянки
Номерні знаки
Ім'я водія
Тарифний план
Час в'їзду

Час виїзду
Вартість простою

Таблиця 2.4.1.

Таблиця 2.4.1. відображає першу таблицю, яка є журналом використання парковки. Елементом чи об'єктом виступає клієнт.

Журнал автостоянки
Номерні знаки
Ім'я водія
Тарифний план
Час в'їзду

Таблиця 2.4.2.

Таблиця 2.4.2. відображає другу таблицю, про увесь транспорт, який в даний момент знаходиться на території автостоянки.

Персональними ідентифікаторами виступають номерні знаки.

ВИСНОВКИ ДО РОЗДІЛУ 2

В даному розділі була розроблена архітектура та модель поставленої задаці, розглянуті окремі компоненти. Було створено концептуальну модель бази даних та розроблена логіка роботи додатку та системи вцілому.

3. ЗАСОБИ РОЗРОБКИ СИСТЕМИ

Враховуючи обраний вид додатку, мовою розробки було обрано Javascript з фреймворком AngularJS, що дозволяє покращити та спростити створення робочого інтерфейсу програми.

Також, одними із використовуваних технологій були обрані такі фреймворки, як Node.js, Bootstrap. Я обрав для роботи саме ці фреймворки тому, що написання коду на них не потрібно вчити нові мови програмування. Усі ці фреймворки, на відміну від інших розробок та технологій, потребують лише знання JavaScript. Використання такого набору інструментів дозволяє створювати сучасні та потужні WEB-програми.

Засоби реалізації при створенні продукту зображені на Рисунку 3.1.

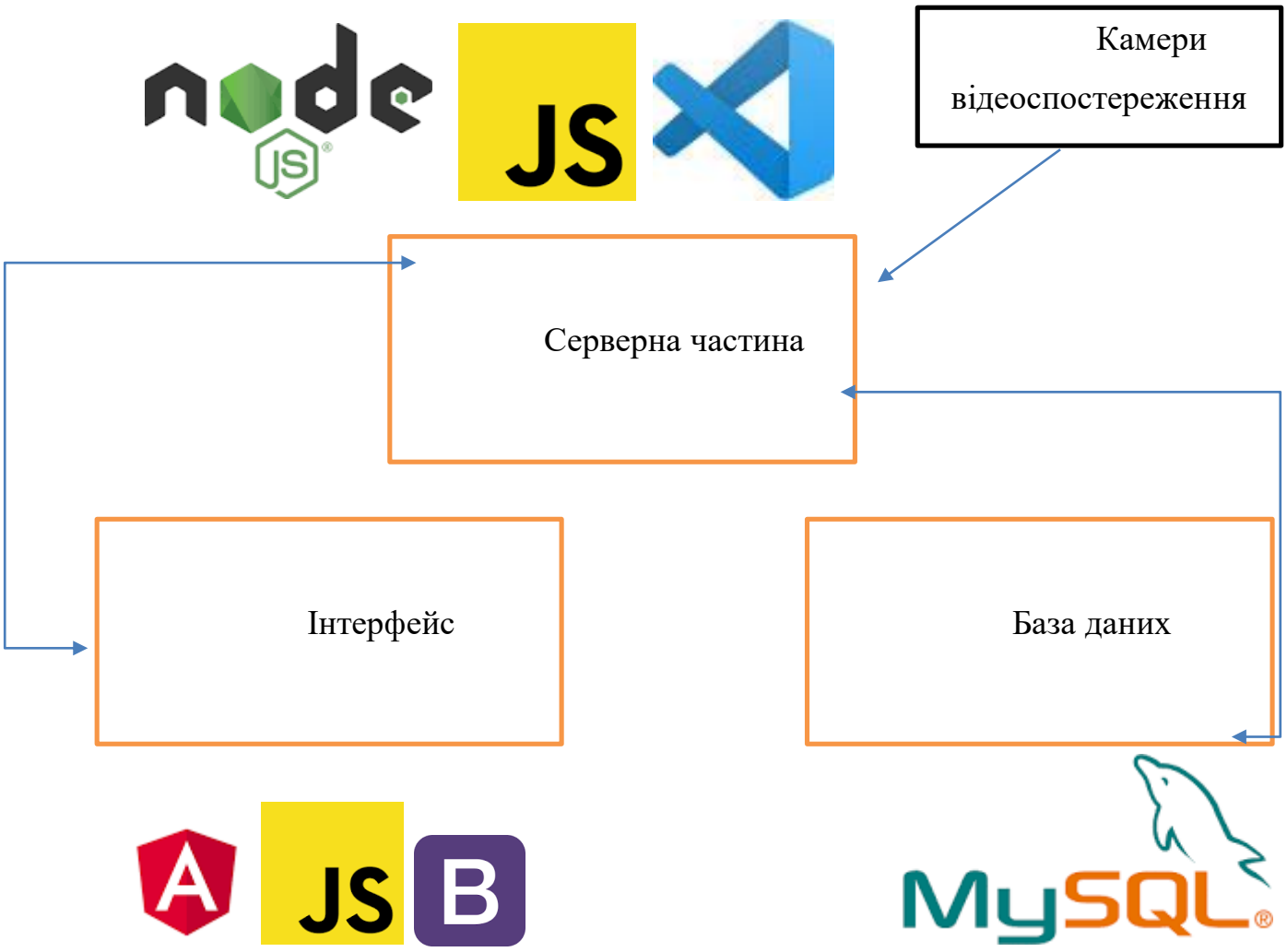

Рисунок 3.1 – набір тенхологій для реалізації програмного забезпечення

Відповідно до Рисунку 3.1 для створення даної системи були використані наступні інструменти та технології (на рисунку відображені лише основні технології для створення прогамного продукту):

- Середовище Visual Studio Code – потужне середовище розробки
- Node.js – програмна платформа для розробки серверної частини
- AngularJS – фреймворк для розробки веб-додатків
- Javascript – основна мова розробки програмного продукту
- Jade – це двигун шаблонів, який використовується в основному для серверних шаблонів у NodeJS.
- MySQL – документоорієнтована система управління базами даних, що не потребує опису схем таблиць
- Bootstrap – фреймворк з додатковими до JavaScript шаблонами та розширеннями
- Git – для контролю версій програмного продукту

Обрані технології дають можливості для повноцінного створення програмного продукту. Середовищем розробки Visual Studio Code було обрано через простоту його інтерфейсу, можливість ставити різноманітні розширення та швидкодію. Основною мовою програмування був обраний Javascript так як вона є найпоширенішою мовою написання WEB-додатків з великою кількустю документацій і розробка серверної частини на Node.js проводилась також на цій мові.

MySQL була обрана як основна база даних через легкість у використанні та простоту інтеграції даної бази з серверною частиною Node.js.

3.1. Середовище розробки Visual Studio Code

Visual Studio Code поєднує простоту редактора вихідного коду з потужним інструментом для розробників, таким як завершення коду IntelliSense та відладка програм.

У редакторі коду встановлені наступлені розширення:

- Angular 2 TypeScript Snippets – для швидкої побудови компонентів Angular коду
- Angular/Bootstrap/Jage/Node.js – основні розширення для використання фреймворків
- Live Sass Compiler – для автоматичного перекомпонування Sass або Scss файлів у CSS
- Bootstrap snippets – для швидкого використання Bootstrap-компонентів
- TSLint – для швидкого покушу рядів помилок та їх виправлення

3.2. Програмна плафторма Node.js

Node.js - це середовище виконання JavaScript з відкритим кодом та міжплатформою. Це популярний інструмент майже для будь-якого проекту! Node.js запускає двигун JavaScript V8, ядро Google Chrome, поза браузером. Це дозволяє Node.js бути дуже ефективним.

Оточення Node.js включає все, що вам потрібно для виконання програми, написаної на JavaScript.

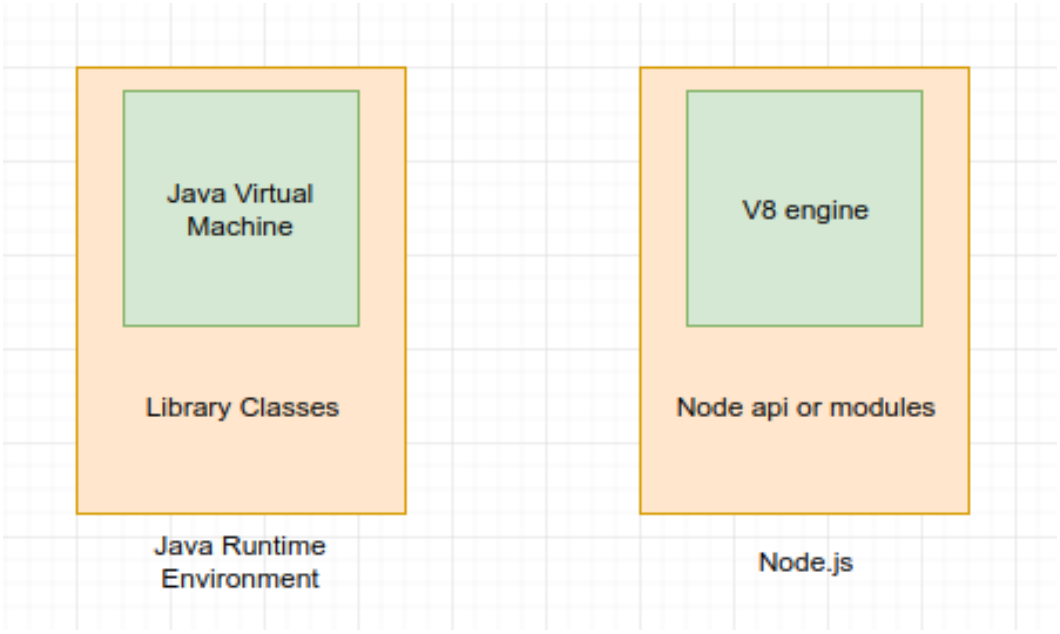

Рисунок 3.2 – проста модель роботи Node.js

Додаток Node.js запускається в одному процесі, не створюючи нової теми для кожного запиту. Node.js надає набір асинхронних примітивів вводу / виводу у своїй стандартній бібліотеці, які запобігають блокуванню коду JavaScript і, як правило, бібліотеки в Node.js записуються з використанням неблокуючих парадигм, що робить поведінку блокування винятком, а не нормою.

Коли Node.js повинен виконати операцію вводу / виводу, як-от читання з мережі, доступ до бази даних або файлової системи, замість блокування потоку і витрачання очікування циклів процесора, Node.js відновить операції, коли відповідь повернеться.

Node.js побудований на основі сучасних версій V8. Отримуючи в курсі останніх випусків цього двигуна, ми гарантуємо, що нові функції зі специфікації JavaScript ECMA-262 своєчасно передаються розробникам Node.js, а також постійні покращення продуктивності та стабільності.

Усі функції ECMAScript 2015 (ES6) розділені на три групи для функцій доставки, інсценування та виконання:

1) Усі функції доставки, які V8 вважає стабільними, увімкнено за замовчуванням на Node.js і не вимагають будь-яких прапорів виконання.
2) Поетапні функції, які є майже завершеними функціями, які команда V8 не вважає стабільними, вимагають прапор часу виконання: --гармонія.
3) У процесі роботи функції можна активувати окремо відповідним прапором гармонії, хоча це вкрай не рекомендується, крім випадків тестування. Примітка: ці прапори піддаються впливу V8 і потенційно можуть змінюватися без будь-якого повідомлення про анулювання.

Команда V8 постійно працює над підвищенням продуктивності нових мовних функцій, щоб з часом досягти паритету зі своїми перекладеними або рідними аналогами в EcmaScript 5 та новіших версіях. Поточний прогрес там відстежується на шестиступінчастому веб-сайті, який показує ефективність функцій ES2015 та ESNext порівняно з їх рідними колегами ES5.

Робота над оптимізацією функцій, впроваджених з ES2015 і далі, координується за допомогою плану продуктивності, де команда V8 збирає та координує ділянки, які потребують вдосконалення, та розробляє документи для вирішення цих проблем.

Наступна діаграма на Рисунку 3.2 зображує деякі важливі частини Node.js:

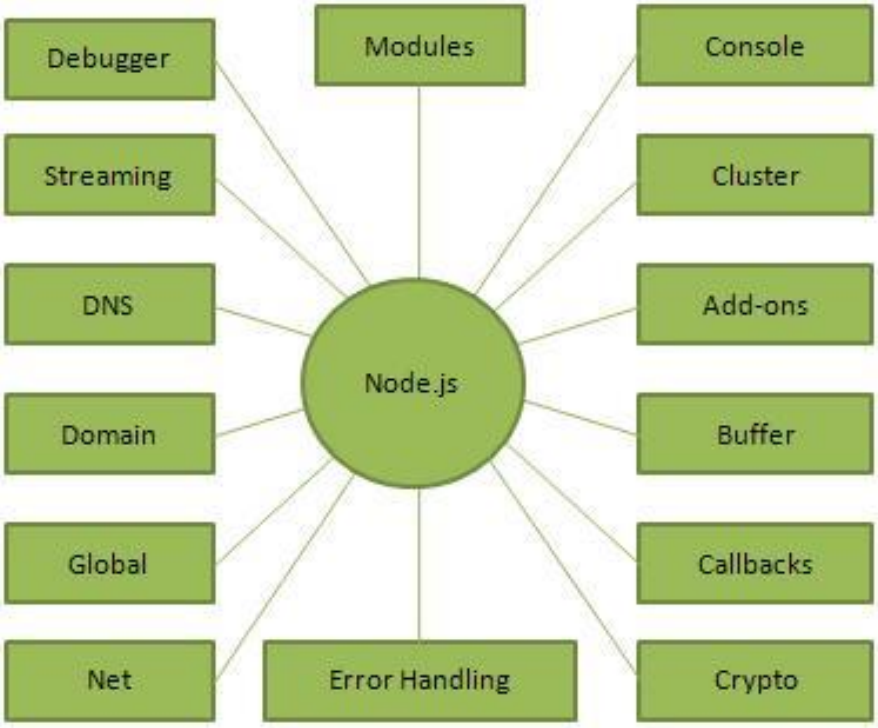

Рисунок 3.2 – різноманітні сфери використання Node.js

Далі йде посилання на github wiki, що містить вичерпний перелік проектів, заявок та компаній, які використовують Node.js. Цей список включає eBay, General Electric, GoDaddy, Microsoft, PayPal, Uber, Wikipins, Yahoo !, та Yammer:

https://github.com/joyent/node/wiki/projects,-applications,-and-companies-using-node

3.3. AngularJS фреймворк

Angular - це одне з найбільш відомих рішень для розробки SPA (односторінковий додаток), окрім React та Vue.js. Це було майже 10 років, і з тих пір

він зазнав незліченну кількість коригувань. Перша версія фреймворку AngularJS почалася ще в 2009 році і поклала початок сучасній розробці прикладних програм.

AngularJS є opensource (програма з відкритим кодом) JavaScript-фреймворк, який використовує шаблон MVC. Власне використання MVC є його однією з відмінних рис.

MVC - Модель-контролер перегляду (MVC) в JavaScript.

Структура Model-View-Controller вимагає деякого опису тут. Назва шаблону складається з імен його дійових осіб: Модель - зберігає дані програми; Перегляд - відображає модель для клієнта; та Controller - оновлює Модель, реагуючи на дії клієнта. Частини архітектури Model-View-Controller визначаються наступним чином:

1) Модель - представлення домену інформації, над якою працює програма. Модель - інша назва доменного шару. Логіка домену додає сенсу необробленим даним (наприклад, обчислення того, чи сьогодні день народження користувача, або підсумків, податків та витрат на доставку товарів для покупок).
2) Перегляд - перетворює модель у форму, придатну для взаємодії, як правило, елемент інтерфейсу користувача. MVC часто спостерігається у веб-додатках, де перегляд - це HTML-сторінка та код, який збирає динамічні дані для сторінки.
3) Контролер - обробляє і реагує на події, як правило, дії користувача, і викликає зміни на моделі та, можливо, в перегляді.

Для опису інтерфейсу використовується декларативне програмування, а бізнес-логіка відокремлена від коду інтерфейсу, що дозволяє поліпшити тестованого і розширюваність додатків.

Іншою відмінною рисою фреймворка є двостороннє зв'язування, що дозволяє динамічно змінювати дані в одному місці інтерфейсу при зміні даних моделі в іншому. Таким чином, AngularJS синхронізує модель і уявлення.

Крім того, AngularJS підтримує такі функціональності, як Ajax, управління структорой DOM, анімація, шаблони, маршрутизація і так далі. Міць фреймворка,

наявність багатого функціоналу багато в чому вплинула на те, що він знаходить своє застосування у все більшій кількості веб-додатків, будучи на даний момент напевно одним з найпопулярніших javascript-фреймворків. На момент написання даного керівництва останньою версією фреймворка була версія 1.6.4.

Angular - це платформа та основа для побудови односторінкових клієнтських додатків за допомогою HTML та TypeScript. Кутовий пишеться в TypeScript. Він реалізує основну та додаткову функціональність як набір бібліотек TypeScript, які ви імпортуєте у свої програми.

Архітектура програми Angular спирається на певні фундаментальні концепції. Основними будівельними блоками є NgModules, які забезпечують контекст компіляції компонентів. NgModules збирають відповідний код у функціональні набори; Кутовий додаток визначається набором NgModules. У додатку завжди є принаймні кореневий модуль, який дозволяє завантажувати, і, як правило, має ще багато функціональних модулів.

Компоненти визначають представлення даних, що представляють собою набори елементів екрану, які можна кутовий вибирати та змінювати відповідно до логіки та даних вашої програми.

Компоненти використовують сервіси, які надають певні функціональні можливості, безпосередньо не пов'язані з переглядами. Постачальники послуг можуть бути введені в компоненти як залежності, що робить ваш код модульним, багаторазовим та ефективним.

І компоненти, і сервіси - це просто класи, в яких є декоратори, які позначають їх тип та надають метадані, які повідомляють Angular, як ними користуватися.

Метадані класу компонентів асоціюють його з шаблоном, який визначає представлення. Шаблон поєднує звичайний HTML з кутовими директивами та розміткою, що дозволяють Angular змінювати HTML, перш ніж відображати його для відображення.

Метадані для класу обслуговування надають інформацію, яку Angular потребує, щоб зробити її доступною для компонентів через введення залежності (DI).

Компоненти програми зазвичай визначають багато представлень, розташованих ієрархічно. Angular надає послугу маршрутизатора, щоб допомогти вам визначити шляхи навігації серед представлень. Маршрутизатор забезпечує складні навігаційні можливості в браузері.

Екземпляр компонента має життєвий цикл, який починається, коли Angular створює компонентний клас і надає перегляд компонента разом із його дочірніми переглядами. Життєвий цикл продовжується з виявленням змін, оскільки Angular перевіряє, коли змінюються властивості, пов'язані з даними, та оновлює як представлення, так і екземпляр компонента за необхідності. Життєвий цикл закінчується, коли Angular знищує екземпляр компонента та видаляє виведений шаблон із DOM. Директиви мають подібний життєвий цикл, як Angular створює, оновлює та знищує екземпляри в процесі виконання.

Наша програма може використовувати методи гачка життєвого циклу, щоб скористатися ключовими подіями життєвого циклу компонента чи директиви, щоб ініціалізувати нові екземпляри, ініціювати виявлення змін, якщо це потрібно, відповідати на оновлення під час виявлення змін та очищати перед видаленням екземплярів.

Ви можете реагувати на події життєвого циклу компонента чи директиви, застосувавши один або кілька інтерфейсів гака життєвого циклу в основній бібліотеці Angular. Гачки дають вам можливість діяти на компонент або екземпляр директиви у відповідний момент, коли Angular створює, оновлює або знищує цей екземпляр.

Кожен інтерфейс визначає прототип для єдиного методу гака, ім'я якого - ім'я інтерфейсу з префіксом ng. Наприклад, інтерфейс OnInit має метод гака, який називається ngOnInit(). Якщо ви реалізуєте цей метод у класі компонентів чи директив, Angular викликає його невдовзі після першої перевірки вхідних властивостей цього компонента чи директиви.

Після того, як ваша програма інстанціює компонент або директиву, викликаючи його конструктор, Angular викликає методи гачка, які ви реалізували у відповідний момент життєвого циклу цього выпадку.

3.4. Мова програмування JavaScript

JavaScript - мова програмування, створена в Netscape як інструмент сценаріїв для маніпулювання веб-сторінками всередині їх браузера, Netscape Navigator.

Частиною бізнес-моделі Netscape був продаж веб-серверів, що включав середовище під назвою Netscape LiveWire, яке могло створювати динамічні сторінки за допомогою сервера JavaScript. На жаль, Netscape LiveWire не був дуже успішним, а JavaScript на сервері до недавнього часу не популяризувався впровадженням Node.js.

Одним із ключових факторів, що призвели до зростання Node.js, були терміни. Лише декількома роками раніше JavaScript почали розглядати як більш серйозну мову завдяки програмам "Web 2.0" (наприклад, Flickr, Gmail тощо), які показали світові, яким може бути сучасний досвід в Інтернеті.

Двигуни JavaScript також стали значно кращими, оскільки багато браузерів змагалися, щоб запропонувати користувачам найкращі показники. Команди розробників, які працюють за основними браузерами, наполегливо працювали, щоб запропонувати кращу підтримку JavaScript та знайти шляхи швидшого запуску JavaScript. Двигун, який Node.js використовує під капотом, V8 (також відомий як Chrome V8, що є двигуном JavaScript з відкритим кодом проекту Chromium Project), значно покращився завдяки цій конкуренції.

В браузері JavaScript може:

-Додавати новий HTML-код на сторінку, змінювати існуючий вміст, модифікувати стилі.

-Реагувати на дії користувача, клацання миші, перемістити вказівник, натискання клавіш.

-Відправляти мережеві запити на віддалені сервера, завантажувати і завантажувати файли (технології AJAX і COMET).

-Отримувати і встановлювати куки, задавати питання відвідувачеві, показувати повідомлення.

-Запам'ятовувати дані на стороні клієнта («local storage»).

JavaScript має три основні переваги над іншими мовами:

1) Повна інтеграція з HTML / CSS.
2) Прості речі робляться просто.
3) Підтримується всіма основними браузерами і включений в них за замовчуванням.

JavaScript - це єдина браузерна технологія, що поєднує в собі всі ці три речі.

Ось що робить JavaScript особливим. Ось чому це найпоширеніший інструмент для створення інтерфейсів в браузері. Хоча, звичайно, JavaScript дозволяє робити програми не тільки в браузерах, але і на сервері, на мобільних пристроях і т.п.

3.5. Bootstrap

Користувацький інтерфейс - це найперша актуальна справа. Тут ми розробляємо простий і привабливий розділ з мінімальними можливостями, використовуючи Bootstrap фреймворк.

Спочатку створений дизайнером та розробником у Twitter, Bootstrap став однією з найпопулярніших фреймворків та проектів з відкритим кодом у світі.

Bootstrap був створений у Twitter в середині 2010 року. До того, як був відкритим джерелом, Bootstrap був відомий під назвою "План Twitter". Кілька місяців після розробки Twitter провів свій перший Hack Week, і проект вибухнув, коли розробники всіх рівнів кваліфікації вскочили без будь-яких зовнішніх вказівок. Він

служив керівництвом стилю для розробки внутрішніх інструментів у компанії понад рік до її публічного виходу, і продовжує це робити і сьогодні.

Спочатку було випущено в п’ятницю, 19 серпня 2011 р. Компанія отримала понад двадцять випусків, включаючи два основні переписувачі з v2 та v3. За допомогою Bootstrap 2 була додана чуйна функціональність до всього фреймворку як факультативний таблицю стилів. Спираючись на Bootstrap 3, ще раз була переписана бібліотеку, щоб зробити її чутливою за замовчуванням за допомогою мобільного першого підходу.

За допомогою програми Bootstrap 4 був знову переписаний проект, щоб врахувати дві ключові архітектурні зміни: міграцію до Sass та перехід до flexbox CSS. Наш намір полягає в тому, щоб допомогти невеликим способом просунути спільноту веб-розробок вперед шляхом висування нових властивостей CSS, меншої кількості залежностей та нових технологій у більш сучасних браузерах.

Bootstrap - це вільна та відкрита рамка для розробки фронтальних версій для створення веб-сайтів та веб-додатків. Рамка Bootstrap побудована на HTML, CSS та JavaScript (JS) для полегшення розробки чуйних сайтів та додатків для мобільних пристроїв.

Чуйний дизайн дає можливість веб-сторінці чи додатку визначати розмір екрана та орієнтацію відвідувача та автоматично адаптувати дисплей відповідно; Мобільний перший підхід передбачає, що смартфони, планшети та спеціальні мобільні додатки - це основний інструмент працівників для роботи і відповідає вимогам цих технологій у дизайні.

Bootstrap включає компоненти користувальницького інтерфейсу, макети та інструменти JS разом із основою для реалізації. Програмне забезпечення доступне заздалегідь складеним або як вихідний код.

Марк Отто та Джейкоб Торнтон розробили Bootstrap у Twitter як засіб для покращення послідовності інструментів, що використовуються на сайті та

скорочення обслуговування. Програмне забезпечення раніше було відоме як Twitter Blueprint, а іноді його називають Twitter Bootstrap.

Основні критерії вибору саме Bootstrap фреймворку для написання приємного інтерфейсу:

1) Легко розпочати
2) Потужна система сіток сторінки
3) Гарні приклади структур та стилів
4) Дужа проста інтеграція
5) Сумісність з усіма браузерами
6) Легке налаштування під себе
7) Постійна підтримка зі сторони розробників
8) Велика кількість плагінів

3.6. MySQL

MySQL - це система управління реляційними базами даних, заснована на SQL - структурованій мові запитів. Додаток використовується для широкого спектру цілей, включаючи програми зберігання даних, електронну комерцію та реєстрацію журналів.

Остання версія MySQL - одна з найпопулярніших баз даних у світі. Це відкритий код, надійний, сумісний з усіма основними хостинг-провайдерами, економічно ефективний і простий в управлінні. Багато організацій використовують безпеку даних та сильну транзакційну підтримку, пропоновану MySQL для захисту онлайн-транзакцій та посилення взаємодії з клієнтами. Однак підприємства, які використовують MySQL, стикаються з кількома проблемами, коли їх додатки відчувають експоненційний ріст і їм потрібні додаткові масштаби.

Поряд з розумінням того, чому MySQL є вирішенням для середовищ із високим ростом, не менш важливо розуміти проблеми, які можуть калічити ваші бізнес-

операції.

Однак найчастіше для mySQL використовується веб-база даних. З його допомогою можна зберігати що завгодно, від одного запису інформації до цілого інвентаря доступних товарів для інтернет-магазину.

5 причин вибору MySQL:

1) Безпечні та захищені транзакції
2) Великий попит
3) Висока доступність
4) Низький поріг входу
5) Швидкий старт роботи

MySQL всесвітньо відомий тим, що є найбільш захищеною і надійною системою управління базами даних, яка використовується в популярних веб-додатках, включаючи WordPress, Drupal, Joomla, Facebook і Twitter. Безпека даних та підтримка транзакційної обробки, яка супроводжує останню версію MySQL, може значною мірою принести користь будь-якому бізнесу, особливо якщо це електронна комерція, яка передбачає часті перекази грошей.

MySQL пропонує неперевершену масштабованість, щоб полегшити управління глибоко вбудованими додатками, використовуючи менший слід, навіть у масивних складах, у яких зберігаються терабайти даних. Гнучкість на вимогу - головна особливість MySQL. Це рішення з відкритим кодом дозволяє повне налаштування для електронної комерції з унікальними вимогами до сервера баз даних.

MySQL має гарантію тривалості роботи 24 × 7 і пропонує широкий спектр високодоступних рішень, включаючи спеціалізовані кластерні сервери.

MySQL вважається однією з дуже швидких мов баз даних, підкріплених великою кількістю тестів та випробовувань

MySQL швидше, надійніше та дешевше через унікальну архітектуру двигуна зберігання. Він забезпечує дуже високопродуктивні результати порівняно з іншими

базами даних, не втрачаючи важливої функціональності програмного забезпечення. Він має утиліти швидкого завантаження через різну пам'ять кешу.

3.7. Jade

Jade - це препроцесор HTML і шаблонізатор, який був написаний на JavaScript для Node.js. Простіше кажучи, Jade - це саме той засіб, який надає вам можливість написання розмітки абсолютно по-новому, з цілим рядом переваг у порівнянні зі звичайним HTML.

3.8. Git

Git - це безкоштовна та з відкритим кодом розповсюджена система управління версіями, призначена для швидкого та ефективного управління всіма проектами від маленьких до дуже великих проектів.

Git - це швидка, масштабована, розподілена система контролю версій з незвичайно багатим набором команд, що забезпечує як операції високого рівня, так і повний доступ до внутрішніх служб.

Git легко засвоїти та має крихітний слід із блискавичною швидкістю. Він випереджає такі засоби SCM, як Subversion, CVS, Perforce та ClearCase з такими функціями, як дешеве місцеве розгалуження, зручні місця постановки та безліч робочих процесів.

ВИСНОВКИ ДО РОЗДІЛУ 3

В третьому розділі був перелічений набір інструментів та технологій для використання у розробці нашого WEB-додатку.

Було обрано мову JavaScript, як основну для написання програми і реализації системи вцілому, так як вона є найбільш розповсюдженою у світі WEB розробки.

Тож, для виконання роботи були обрані найзручніші та новіші технології.

4. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ДОДАТКОМ

4.1. Послідовність дій при роботі

Для роботи користувачу необхідно мати сам програмний додаток на робочому ПК на робочому місці.

Після того як користувач переходить на WEB-додаток, він повинен авторизуватися під своїм робочим логіном на паролем.

Приклад роботи на рисунку 4.1.

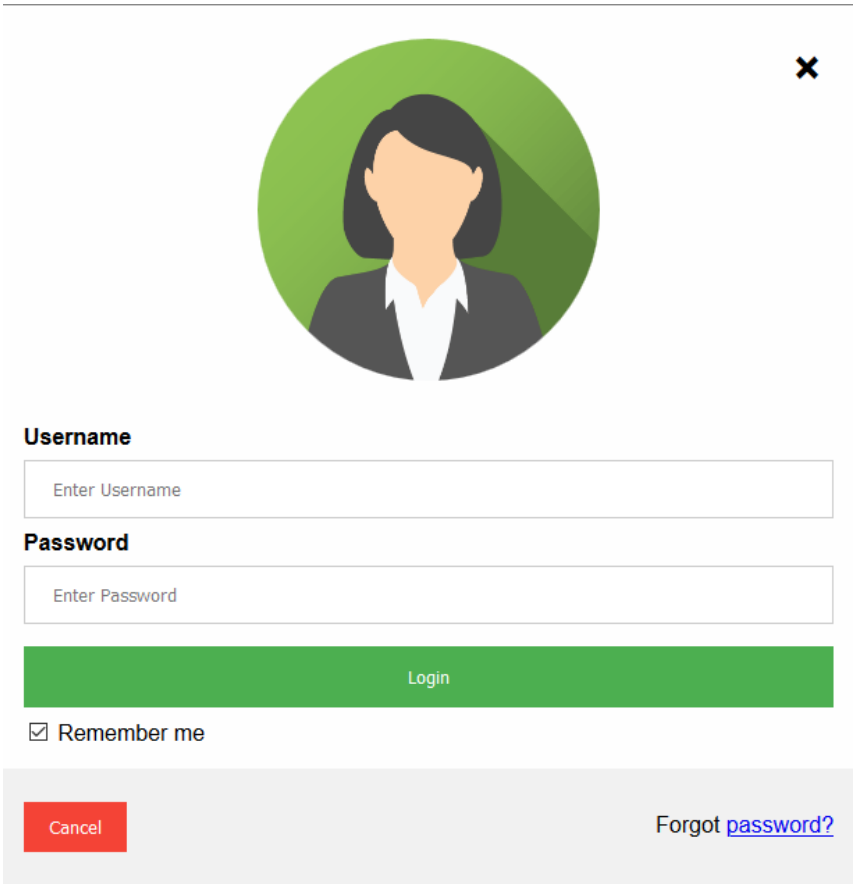

Рисунок 4.1.

Програма розпізнає 2 види облікових записів:

- адміністратор
- оператор

Адміністратор має повний набір інструментів для управління системою, таких як видалення клієнта з історії, зміна тарифного плану клієнта, управління обліковими записами усіх операторів системи, додавання та видалення.

Оператор у свою чергу має менший функціонал і може працювати лише з даними про автомобільний транспорт.

Працівник	Адміністратор	Оператор
Опис	Має доступ до бази даних, може змінювати права доступу для первних операторів, додавати, видаляти дані про клієнтів, може переглядати або видаляти історію подій на автостоянці.	Має доступ до бази даних, може встановлювати тарифний план, проте не може редагувати персональні дані у таблиці клієнтів.
Відповідальність	Контролює правильність та доцільність тарифів та слідкує за доброчесністю оператора	Контролює правильність вводу системою номерних знаків, підтверджує факт проїзду машини до парковки

Далі після запуску користувач бачить перед собою історію відвідувань автостоянки та пряму трансляцію з камер відеоспостереження, та може переглядати дані по всім машинам, які були клієнтами.

Коли до парковки під’їжає нова машина камери відеоспостереження зчитують номерні знаки та оператору висвічується вибір тарифного плану для даної машини:

- Постійний клієнт
- VIP-клієнт

- Звичайний клієнт

Якщо в базі даних машина не зазначена як VIP чи постійний клієнт, оператор обирає «звичайний клієнт» і піднімає шлагбаум.

В цей момент в базу даних записується час прибуття, номерні знаки, кількість відвідувань (для подальшого перетворення на постійного клієнта) та по бажанню директора/начальника парковки персональні дані клієнта.

Також бувають ситуації, коли камера відеоспостереження не може розпізнати номерні знаки через погодні умови або забрудненість номерів. В такій ситуації програма надає оператору можливість ввести дані власноруч.

При виїзді машини з автостоянки камери знову зчитують номерні знаки, та відправляють дані на сервер разом з часом виїзду і на на сервері підраховується вартість за простій.

4.2. Системні вимоги

Наявність комрьютера з будь-яким браузером та встановлене на ньому програмне забезпечення:

1) Фреймворк AngularJS
2) Фреймворк Node.JS
3) Фреймворк Express.JS
4) База даних для зберігання історії парковки
5) WEB-браузер для роботи з системою

ВИСНОВОК

В ході виконаної роботи був отриманий великий досвід у роботі з розробкою клієнт-серверних програм з використанням моделі REST. Відповідно до завдання було створено WEB-додаток, що дозволяє контролювати та обслуговувати пропускну здатність на парковках та стоянках.

Сформульована проблема створення системи для покращення пропускної здотності до автостоянок.

Невід’ємною частиною роботи було створення серверу, яких робить основні підрахунки та зв’язує WEB-сторінку з базою даних, в якій містяться дані про усі автомобілі, їх власників, дату, тариф та час перебування на території парковки.

Дана задача розширила знання роботи з такими технологіями як AngularJs, Node.js, Bootstrap, MySQL.

Додаток успішно справляється з поставленими перед ним задачами.

Отже, створення та розробка даногї системи покращило навички роботи з різноми технологіями, які існують у сучасному світі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Сайт Jade Tutorial [Електронний ресурс] — Режим доступу: https://habr.com/ru/post/278109/

2. Сайт Bootstrap features [Електронний ресурс] — Режим доступу: https://www.creative-tim.com/blog/web-design/key-features-look-bootstrap-template/

3. Сайт intteks [Електронний ресурс] — Режим доступу: http://intteks.com.ua/products/sistemy-kontrolya-transporta/sistema-avtomatizirovannoj-parkovki-cargo

4. Сайт Node JS [Електронний ресурс] — Режим доступу: https://marketplace.visualstudio.com/items?itemName=Zaczero.bootstrap-v4-snippets

5. Сайт Bootstrap [Електронний ресурс] — Режим доступу: https://ng-bootstrap.github.io/#/home

6. Сайт Основи MySQL[Електронний ресурс] — Режим доступу: https://wiki.gentoo.org/wiki/MySQL/Startup_Guide/ru

7. Сайт Node JS [Електронний ресурс] — Режим доступу: https://medium.com/devschacht/node-hero-chapter-1-239f7afeb1d1

8. Сайт AngularJS lifecycle [Електронний ресурс] — Режим доступу: https://angular.io/guide/lifecycle-hooks

9. Сайт JavaScript [Електронний ресурс] — Режим доступу: https://learn.javascript.ru/

10. Сайт Javascript Features [Електронний ресурс] — Режим доступу: http://simplyaccessible.com/article/the-accessibility-stack

11. Сайт Основи Jade [Електронний ресурс] — Режим доступу: http://jade-lang.com/

12. Стаття про REST [Електронний ресурс] — Режим доступу: https://medium.com/extend/what-is-rest-a-simple-explanation-for-beginners-part-1-introduction-b4a072f8740f

13. Стаття про MVC [Електронний ресурс] — Режим доступу: https://alexatnet.com/model-view-controller-mvc-in-javascript/

Додаток 1

Розробка програмного комплексу для збору та аналітики даних системи управління та контролю доступу

Специфікація

УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТМ6174_20Б

Аркушів 1

Київ — 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТМ61174_20Б	ТМ-61 Литвиненко К.А. бакалавр.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”ІМ.ІГОРЯ_ СІКОРСЬКОГО_ТЕФ_АПЕПС_ТМ61174_20Б 12-1	Bachelors.rar	Основний файл архіву, в якому лежать файли програми та необхідні бібліотеки
УКР.НТУУ”КПІ”ІМ.ІГОРЯ_ СІКОРСЬКОГО_ТЕФ_АПЕПС_ТМ61174_20Б 13-1	Опис.docx	Опис програмного модулю

Додаток 2

Розробка програмного комплексу для збору та аналітики даних системи управління та контролю доступу

Текст програмного модулю

УКР.НТУУ”КПІ”ІМ.ІГОРЯ_ СІКОРСЬКОГО_ТЕФ_АПЕП С_ТМ61174_20Б 12-1

Аркушів 7

Київ — 2020

```
var createError = require('http-errors');
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');
const mongoose = require('mongoose');
var bodyParser = require('body-parser');

var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');

var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');

app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(bodyParser.urlencoded({
  extended: true
}));
app.use(express.static(path.join(__dirname, 'public')));
```

```
app.use('/', indexRouter);
app.use('/users', usersRouter);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  next(createError(404));
});

mongoose.connect('mongodb+srv://kek:123qwe@dbford-
mqmh5.mongodb.net/shop?retryWrites=true&w=majority', {
  useNewUrlParser: true,
  useUnifiedTopology: true
});
mongoose.set('useFindAndModify', false);
const db = mongoose.connection;
db.on('error', console.error.bind(console, 'connection error:'));
db.once('open', function() {
  // we're connected!
  console.log('connected to db!')
});

// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
```

```
});

module.exports = app;
var express = require('express');
var router = express.Router();
const itemsModel = require('../models/items');
const mongoose = require('mongoose');

/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Express' });
});

router.post('/add',function(req,res,next){
  console.log(req.body);
  let items = new itemsModel({
      name: req.body.name,
      start: req.body.start,
      sign: req.body.sign,
      desc: req.body.desc,
  }).save();
  res.send(items);
});

router.get('/items', function(req,res,next){
  let newparam = [];
  let prom1;
  let prom2;
  let prom3;
  let prom5;
```

```
        console.log("queries: ",req.query);
        if(!req.query.category && !req.query.min && !req.query.max &&
!req.query.id){
          console.log('no params');
          prom3 = itemsModel.find({})
          .limit(20)
          .then((data)=>{
            for(const dota in data){
              newparam.push(data[dota]);
            }
          })
        }

        if(req.query.name && req.query.name !=="all"){
          prom5 = itemsModel.find({name:req.query.name})
          .then((data)=>{
            if(req.query.min && req.query.max){
              console.log("got prices");
              for(const i in data){
                if(data[i].price > req.query.min && data[i].price < req.query.max){
                  newparam.push(data[i])
                }
              }
            }else if(req.query.min && !req.query.max){
              for(const i in data){
                if(data[i].price > req.query.min){
                  newparam.push(data[i])
                }
              }
            }else if(req.query.max && !req.query.min){
```

```
    for(const i in data){
      if(data[i].price < req.query.max){
        newparam.push(data[i])
      }
    }
  }else if(!req.query.min && !req.query.max){
    for(const i in data){
      newparam.push(data[i]);
    }
  }
 })
}

if(req.query.category =="all" || !req.query.category){
 prom5 = itemsModel.find()
 .then((data)=>{
  if(req.query.min && req.query.max){
    console.log("got prices");
    for(const i in data){
      console.log(data[i].price);
      if(data[i].price > req.query.min && data[i].price < req.query.max){
        newparam.push(data[i])
      }
    }
  }else if(req.query.min && !req.query.max){
    for(const i in data){
      if(data[i].price > req.query.min){
        newparam.push(data[i])
      }
    }
```

```
    }else if(req.query.max && !req.query.min){
      for(const i in data){
        if(data[i].price < req.query.max){
          newparam.push(data[i])
        }
      }
    }else if(!req.query.min && !req.query.max){
      for(const i in data){
        newparam.push(data[i]);
      }
    }
  })
}


let categoriesArray = [];
let getCategories = itemsModel.find()
  .then((data)=>{
    for(const dota in data){
      categoriesArray.push(data[dota].name);
    }
  })
Promise.all([prom1, prom2, prom3,getCategories, prom5])
  .finally(() => {
    let clearDuplicates = (names) => names.filter((v,i) => names.indexOf(v) === i)
    let clearCagetories = clearDuplicates(categoriesArray);


res.render('items',{items:newparam,categories:clearCagetories,countItems:newparam.lengt
h});
```

```
    })

})
module.exports = router;
```

Додаток З

Розробка програмного комплексу для збору та аналітики даних системи управління та контролю доступу

Опис програмного модулю

УКР.НТУУ”КПІ”ІМ.ІГОРЯ_ СІКОРСЬКОГО_ТЕФ_АПЕП С_ТВ61174_20Б 13-1

Аркушів 9

Київ — 2020

АНОТАЦІЯ

Метою роботу було створення програмного комплексу для збору та аналітики даних системи управління та контролю доступу.

Було створено клієнтський додаток за допомогою мови C# у інтегрованій середі розробки Microsoft Visual Studio за використанням Windows Forms та працює з базою даних розробленою на платформі MySQL.

Додаток дозволяє збирати, записувати у базу даних, аналізувати дані доступу до різних приміщень.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У цьому додатку міститься опис програмної системи для контролю пропускної здатності до автостоянки та тарифікації послуг.

Додаток працює в будь якому бразуері, і потребує встановлених на ПК фреймворків та бази даних MySQL.

При розробці програмного додатку використовувалась мова JavaScript з використанням середовища Microsoft Visual Studio Code.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений додаток виконує завдання контролю доступу автомобілів до стоянки та тарифікації послуг. Дані які вводяться користувачем, проходять перевірку на правильність.

Всі данні користувачів імпортуються з бази даних, та всі події, та також записуються в базу даних. Також нові користувачі можуть бути мануально додані до бази даних.

Всі дані входів\виходів можуть бути проаналізовані додатково, якщо буде необхідно.

Проблему вирішено завдяки мові програмування JavaScript. Також для логування події використана база даних MySQL.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

При запуску інформаційної системи користувач бачить початкове вікно програми, з якого може вводити відповідні дані входу\виходу, передивлятися лог подій, корегувати або видаляти.

Логічна структура програми складається з методів та функції, які виконують поставлені задачі. Програма містить директорію компонентів, де знаходяться спільні компоненти для користувацького інтерфейсу. Також існують допоміжні модулі, які формують структуру програми — це конфіги, допоміжні функції тощо.

Інтерфейс користувача – виконано з використанням таких технологій:

- мова JavaScript - для написання логіки додатку;
- бібліотеки Angular та Bootstrap, що надають інструментарій для візуалізації інформації у графічному вигляді;
- база даних MySQL для зберігання даних нашої системи.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для ствворення системи були використані засоби мови програмування JavaScript та бібліотеки цієї мови.

Задачі візуалізації інтерфейся і зберігання даних були вирішені за допомогою AngularJS та бази даних MySQL.

Написання коду відбувалося в Microsoft Visual Studio Code, та база даних розташовується на локальному сервері localhost.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Розроблений модуль не потребує інсталяції, треба запустити сервер та базу даних в командному рядку та перейти на сайт у браузері.

Після запуску програми користувач переходить до початкового меню авторизації, звідки може почати роботу та виконувати потрібні дії.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними для розроблених додатків є інформація яка зчитується з камер відеоспостереження MySQL та дані введені в вікні програми.

Вхідні дані зберігаються у програму та представлені у вигляді об’єктів класів, юзерів, евентів тощо.

Вихідними даними є дані в’їзду\виїзду користувачів, які відображаються у головному вікні програми та записуються у базу даних.